

AD-A114 923

HARVARD UNIV CAMBRIDGE MA AIKEN COMPUTATION LAB
O (LOS N) TIME RECOGNITION OF DETERMINISTIC CLFS.(U)
APR 82 J H REIF

F/G 5/7

N00014-80-C-0674

UNCLASSIFIED

TR-10-82

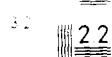
NL

1 OF 1
ALIA
1-8-82

END
DATE
FILMED
06-82
DTIC



2.8 2.5



2.0



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A 11 4923

DTIC FILE COPY

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A114923	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) O(log N) Time Recognition of Deterministic CLFs		5. TYPE OF REPORT & PERIOD COVERED Technical Report
7. AUTHOR(s) John H. Reif		6. PERFORMING ORG. REPORT NUMBER TR-10-82
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harvard University Cambridge, MA 02138		8. CONTRACT OR GRANT NUMBER(s) N00014-80-C-0674
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research 800 North Quincy Street Arlington, VA 22217		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) same as above		12. REPORT DATE April, 1982
		13. NUMBER OF PAGES 21
		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) unlimited <div style="border: 1px solid black; padding: 5px; display: inline-block;"> This document has been approved for public release and sale; its distribution is unlimited. </div>		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) unlimited		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) language recognition, parallel algorithm, parallel RAM, deterministic pushdown machines, deterministic context free languages.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) see reverse side		

DTIC
ELECTE
MAY 26 1982
H

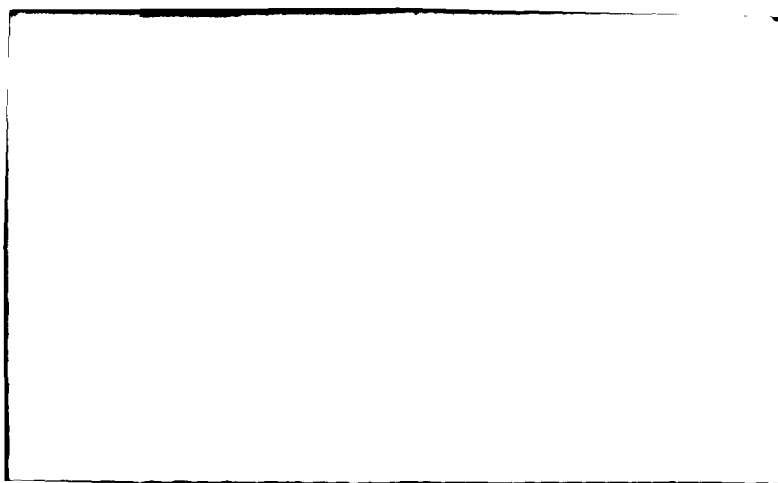
DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

SUMMARY. We prove that the languages accepted by auxiliary deterministic pushdown machines with space $s(n) \geq \log n$ and time $2^{O(s(n))}$ are accepted in time $O(s(n))$ by parallel RAMs. Thus deterministic context free languages are accepted in time $O(\log n)$ and a polynomial number of processors by parallel RAMs. Also we show that the languages accepted in time $T(n)$ by parallel RAMs are accepted with simultaneous space $T(n)$ and time $2^{O(T(n)^2)}$ by auxiliary deterministic pushdown machines.



$O(\log N)$ Time Recognition of Deterministic CFLs

John H. Reif

TR-10-82

April, 1982

DTIC
Electronics
MAY 26 1982

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

O(log N) Time Recognition of Deterministic CFLs

John Reif*

Aiken Computation Laboratory
Division of Applied Sciences
Harvard University
Cambridge, Massachusetts

SUMMARY. We prove that the languages accepted by auxiliary deterministic pushdown machines with space $s(n) \geq \log n$ and time $2^{O(s(n))}$ are accepted in time $O(s(n))$ by parallel RAMs. Thus deterministic context free languages are accepted in time $O(\log n)$ and a polynomial number of processors by parallel RAMs. Also, we show that the languages accepted in time $T(n)$ by parallel RAMs are accepted with simultaneous space $T(n)$ and time $2^{O(T(n)^2)}$ by auxiliary deterministic pushdown machines.



Accession For	DTIC	DTIC	DTIC
NTIS	GR&I		
DTIC	IAB		
Unannounced			
Justification			
By			
Distribution/			
Availability Codes			
Avail and/or			
Dist			
Special			

A

* This work was supported in part by National Science Foundation Grant NSF-MCS79-21024 and the Office of Naval Research Contract N00014-80-C-0647.

1. INTRODUCTION

This paper assumes the parallel random access machine model *P-RAM* as defined in [Fortune and Wyllie, 78] and [Wyllie, 79], which consists of a collection of synchronous deterministic unit-cost RAMs with shared memory locations indexed by the natural numbers. Simultaneous reads are allowed but no two distinct processors can attempt to simultaneously write into the same memory location at the same time.

Previously [Fortune and Wyllie, 78] showed (see Section 3) that any language accepted by a deterministic Turing machine with space bound $s(n) \geq \log n$, is accepted in time $O(s(n))$ by a P-RAM. Also [Ruzzo, 79] showed that any language accepted by an auxiliary pushdown machine with space $s(n) \geq \log n$ and time $2^{O(s(n))}$ is accepted in time $O(s(n))^2$ in various parallel machine models including the P-RAM.

Section 4 of this paper gives a $O(s(n))$ time P-RAM algorithm for recognizing the language $L(M)$ of an auxiliary pushdown machine M with space $s(n) \geq \log n$ and $2^{O(s(n))}$ time. In the case L is a deterministic CFL, then our algorithm takes time $O(\log n)$ and requires $n^{O(1)}$ processors.

Section 5 proves the correctness of our algorithm. We use arguments somewhat similar to those of [Cook, 79] for acceptance of deterministic CFLs in simultaneously polynomial time and log squared space.

Section 6 proves a complementary result where we also prove that any language accepted by a deterministic space $T(n)$ bounded P-RAM is accepted by a space $T(n)$, $2^{\alpha(T(n)^2)}$ time auxiliary deterministic push-down machine.

Section 7 describes some further results including improvement of processor bounds, parallel recognition of the LR(k) languages, and also implementation of our algorithm on the HMMs of [Cook, 80].

2. PRELIMINARY DEFINITIONS AND ASSUMPTIONS

Consider a fixed deterministic auxiliary pushdown machine (APDA) M with work space bound $s(n) \geq \log n$ and $2^{O(s(n))}$ time for an input string of length n .

Let a *position* π be a finite string containing:

- (i) the current state of M (in the finite control)
- (ii) the position of the input head
- (iii) the contents of the work tapes and positions of the work tape heads.

Let Π be the set of positions of M with work space $s(n)$. Let an *instantaneous description* (ID) of M be a pair (π, σ) where π is a position in Π and σ is the current stack contents.

Let $NEXT$ be the *next move function* of M be defined as usual for APDA (see for example [Aho, Hopcroft, and Ullman, 79]) such that for each ID (π, σ) , $(\pi', \sigma') = NEXT(\pi, \sigma)$ is the ID immediately following (π, σ) ; let $\pi' = PNEXT(\pi, \sigma)$ be the resulting position and let $\sigma' = SNEXT(\pi, \sigma)$ be the resulting stack contents. Let $LOOP(\pi, \sigma)$ be the predicate that is *true* iff $(\pi, \sigma) = NEXT(\pi, \sigma)$.

For each ID (π, σ) , let $COMP(\pi, \sigma)$ be the sequence of IDs

$$(\pi = \pi_0, \sigma = \sigma_0), (\pi_1, \sigma_1), \dots$$

such that $(\pi_i, \sigma_i) = NEXT(\pi_{i-1}, \sigma_{i-1})$ for $i = 1, 2, \dots$ and let $PCOMP(\pi, \sigma)$ be the sequence of positions π_0, π_1, \dots .

Let Σ be the finite input alphabet of M . Given input string $w \in \Sigma^n$, let $(\pi_I(w), \lambda)$ be the *initial ID* and let (π_A, λ) be the *accepting ID*, where λ is the empty stack. M *accepts* w iff (π_A, λ) is in $COMP(\pi_I(w), \lambda)$.

To simplify the presentation of our recognition algorithm, we make, without loss of generality, the following *assumptions*:

- A_1 There is always a next move from any ID, and M never attempts to pop the empty stack λ .
- A_2 Only pop moves are dependent on the value at the top of the stack, if the stack is nonempty.
- A_3 If $LOOP(\pi, \sigma)$ then $\sigma = \lambda$ (i.e., M must empty the stack before looping).
- A_4 $LOOP(\pi_A, \lambda)$ (i.e., M loops at the accepting ID (π_A, λ)).
- A_5 Each position $\pi \in \Pi$ has two counters containing numbers $t(\pi), h(\pi) \geq 0$ where
 - (i) $t(\pi_I(w)) = h(\pi_I(w)) = 0$ for the initial ID and
 - (ii) for each ID (π, σ) , if $(\pi', \sigma') = NEXT(\pi, \sigma)$ and $LOOP(\pi, \sigma) = \text{false}$ then $t(\pi') = t(\pi) + 1$ and $h(\pi') = h(\pi) + |\sigma'| - |\sigma|$ (so $h(\pi') = h(\pi) - 1$ if the move is a pop, $h(\pi') = h(\pi) + 1$ if the move is a push, and otherwise $h(\pi') = h(\pi)$).

Note that assumption A_5 requires only additional work space $2s(n)$.

PROPOSITION 2.1. \exists constant $c > 0$ such that $|\Pi| \leq c^{s(n)}$ for all $n \geq 0$.

For each $t \geq 0$ and each ID (π, σ) , let

$$COMP_t(\pi, \sigma) = \{(\pi', \sigma') \in COMP(\pi, \sigma) \mid t(\pi') \geq t \text{ or } LOOP(\pi', \sigma')\}$$

and let $PCOMP_t(\pi, \sigma) = \{\pi' \mid (\pi', \sigma') \in COMP_t(\pi, \sigma)\}$.

PROPOSITION 2.2 \exists constant $c_1 \geq 0$ such that M accepts w iff

$$COMP_{c_1 s(n)}(\pi_I(w), \lambda) = \{(\pi_A, \lambda)\}.$$

The following elementary propositions will be of use.

PROPOSITION 2.3. If $(\pi', \sigma') \in \text{COMP}(\pi, \lambda)$ and $(\pi'', \sigma'') \in \text{COMP}(\pi', \lambda)$ then $(\pi'', \sigma' \circ \sigma'') \in \text{COMP}(\pi, \lambda)$. (See Figure 1.)

PROPOSITION 2.4. If $(\pi', \lambda) \in \text{COMP}(\pi, \lambda)$ and $(\pi'', \sigma'') \in \text{COMP}(\pi', \sigma)$ then $(\pi'', \sigma'') \in \text{COMP}(\pi, \sigma)$. (See Figure 2.)

3. PARALLEL SIMULATION OF DETERMINISTIC TMS

This section considers the special case where the deterministic APDA M never pushes a symbol onto its stack; thus M is essentially a *deterministic TM* (the techniques used in this case are generalized in the next section). We use a known parallel algorithm for this case, due to [Fortune and Wyllie, 78] and [Wyllie, 79]. We assume M has input $w \in \Sigma^n$, space $s(n) \geq \log n$ and let $s_1(n) = \lceil s(n) \log c_1 \rceil$, where the constant c_1 is as given in Proposition 2.2. For succinctness we drop in this section the second argument (i.e., the stack which will always be λ) to PNEXT , PCOMP , and LOOP .

To initialize let $P_0(\pi) = \text{PNEXT}(\pi)$ for each position $\pi \in \Pi$. Then for each stage $k = 0, 1, \dots, s_1(n) - 1$ let $P_{k+1}(\pi) = P_k(P_k(\pi))$ simultaneously for each position $\pi \in \Pi$. (See Figure 3.) *Accept* input w iff

$$P_{s_1(n)}(\pi_I(w)) = \pi_A.$$

By Proposition 2.1, we can encode each position $\pi \in \Pi$ as a number $\langle \pi \rangle$ of size $O(s(n))$. For each $\pi \in \Pi$ a single processor is used to compute $P_k(\pi)$ (and write $P_k(\pi)$ into a memory location indexed by $\langle \pi \rangle$) at stages $k = 0, 1, \dots, s_1(n)$. Note that each stage can be done in a constant number of P-RAM steps, using the ability of processors to synchronously write and then read from memory cells indexed by a registers. (For details see [Fortune and Wyllie, 79]). By Proposition 2.1, $|\Pi| = 2^{O(s(n))}$. Therefore:

THEOREM 3.1. A P-RAM can compute $P_{s_1(n)}(\pi_I(w))$ in time $O(s(n))$ and $2^{O(s(n))}$ processors.

The correctness of this algorithm follows from:

THEOREM 3.2. For each stage $k \geq 0$ and position $\pi \in \Pi$,

$P_k(\pi) \in \text{PCOMP}_{2^k+t(\pi)}(\pi)$. (Note thus $P_k(\pi)$ is reachable from π by at least 2^k moves.)

Proof by induction on k . Suppose the theorem holds for a fixed k .

Then by the induction hypothesis $P_k(\pi) \in \text{PCOMP}_{2^k+t(\pi)}(\pi)$ and $P_{k+1}(\pi) = P_k(P_k(\pi)) \in \text{PCOMP}_{2^k+t(P_k(\pi))}(P_k(\pi))$. By Proposition 2.3, $P_{k+1}(\pi) \in \text{PCOMP}_{t_0}(\pi)$ for some $t_0 = 2^k + t(P_k(\pi))$. If $\text{LOOP}(P_k(\pi))$, then $P_{k+1}(\pi) = P_k(\pi)$ so $\text{LOPP}(P_{k+1}(\pi))$. Otherwise, $t(P_k(\pi)) \geq 2^k + t(\pi)$ so $t_0 \geq 2^k + (2^k + t(\pi)) = 2^{k+1} + t(\pi)$. Thus in either case, $P_{k+1}(\pi) \in \text{PCOMP}_{2^{k+1}+t(\pi)}(\pi)$. \square

COROLLARY 3.2. $P_{s_1(n)}(\pi_I(w)) = \pi_A$ iff M accepts w .

Proof. By Theorem 3.2, $P_{s_1(n)}(\pi_I(w)) \in \text{PCOMP}_{c_1 s(n)}(\pi_I(w))$. But by Proposition 2.2., $\text{PCOMP}_{c_1 s(n)}(\pi_I(w)) = \{\pi_A\}$ iff M accepts w . \square

4. PARALLEL SIMULATION OF A DETERMINISTIC APDA

This section gives our parallel algorithm for simulating a deterministic APDA M with space $s(n) \geq \log n$. Let $w \in \Sigma^n$ be the input string and again let $s_1(n) = \lceil s(n) \log c_1 \rceil$.

To initialize for each position $\pi \in \Pi$, we let

$$P_0(\pi) = \text{PNEXT}(\pi, \lambda)$$

$$L_0(\pi) = \begin{cases} P_0(\pi) & \text{if } h(P_0(\pi)) = 0 \\ \pi & \text{else} \end{cases}$$

Also, for each $\pi, \pi' \in \Pi$ such that $t(\pi') \geq t(P_0(\pi))$ and $h(\pi') \geq h(\pi)$, we let

$$\text{PREDICT}_0(\pi, \pi') = \begin{cases} \pi'' & \text{if } h(\pi') = h(P_0(\pi)) \geq h(\pi'') \\ \pi' & \text{else} \end{cases}$$

where $\pi'' = \text{PNEXT}(\pi', \text{SNEXT}(\pi, \lambda))$

and let $Q_0(\pi) = \text{PREDICT}_0(\pi, P_0(\pi))$.

For each $k = 0, 1, \dots, s(n)-1$ and each position $\pi \in \Pi$ we let

$$P_{k+1}(\pi) = P_k(Q_k(\pi)) \quad (\text{see Figure 4})$$

$$L_{k+1}(\pi) = \begin{cases} L_k(Q_k(\pi)) & \text{if } h(Q_k(\pi)) = h(\pi) \\ L_k(\pi) & \text{else} \end{cases} \quad (\text{see Figure 5})$$

$$Q_{k+1}(\pi) = \text{PREDICT}_{k+1}(\pi, P_{k+1}(\pi))$$

as defined below.

For each $\pi, \pi' \in \Pi$ where $t(\pi') \geq t(\pi)$ and $h(\pi') \geq h(\pi)$ let

$$\text{HOP}_{k+1}(\pi, \pi') = \begin{cases} \pi_2 & \text{if } h(Q_k(\pi_1)) = h(\pi_1) \\ L_k(\pi_1) & \text{else} \end{cases}$$

where

$$\pi_1 = \text{PREDICT}_k(\pi, \pi') \quad \text{and} \quad \pi_2 = \text{PREDICT}(\pi, Q_k(\pi_1))$$

(see Figure 6)

For each $\pi, \pi' \in \Pi$ such that $t(\pi') \geq t(P_{k+1}(\pi))$ and $h(\pi') \geq h(\pi)$

let

$$\text{PREDICT}_{k+1}(\pi, \pi') = \begin{cases} \hat{\pi} & \text{if } h(\hat{\pi}) = h(Q_k(\pi)) \\ \hat{\pi} & \text{else} \end{cases}$$

where

$$\hat{\pi} = \text{HOP}_{k+1}(Q_k(\pi), \pi') \quad \text{and} \quad \hat{\pi} = \text{HOP}_{k+1}(\pi, \hat{\pi}) \quad (\text{see Figure 7}).$$

Finally, we accept input w iff $P_{s_1(n)}(\pi_I(w)) = \pi_A$.

THEOREM 4.1. A P-RAM can compute $P_{s_1(n)}(\pi_I(w))$ in time $O(s(n))$ and $2^{O(s(n))}$ processors.

Proof. Let $m = |\Pi|$. By Proposition 2.1, $m = 2^{O(s(n))}$. We devote a processor for each $\pi \in \Pi$ to synchronously compute $P_k(\pi)$, $L_k(\pi)$, and $Q_k(\pi)$ (and write these values into memory locations indexed by $\langle \pi \rangle$, $\langle \pi \rangle + m$, $\langle \pi \rangle + 2m$, respectively) at stages $k = 0, 1, \dots, s_1(n)$. Also we use a processor for each pair $\pi, \pi' \in \Pi$ to synchronously compute first $\text{HOP}_k(\pi, \pi')$ and then $\text{PREDICT}_k(\pi, \pi')$ (and write these values into memory locations indexed by $\langle \pi \rangle + m(3 + \langle \pi' \rangle)$ and $\langle \pi \rangle + m(3 + m + \langle \pi' \rangle)$, respectively) at stages $k = 1, \dots, s_1(n)$.

As in the proof of Theorem 3.1, each stage takes only a constant number of P-RAM steps. □

5. PROOF OF CORRECTNESS

This section proves that for our algorithm presented in Section 4,

THEOREM 5.1. $P_k(\pi) \in \text{PCOMP}_{2^{k+t(\pi)}}(\pi, \lambda)$ for each position $\pi \in \Pi$ and stage $k \geq 0$.

By Proposition 2.2 we have:

COROLLARY 5.1. $P_{s_1(n)}(\pi_I(w)) = \pi_A$ iff M accepts w .

Although our algorithm of the previous section does not actually compute a push-down stack, it is useful in our proof of Theorem 5.1 to introduce variables denoting the stack contents at various stages of the deterministic APDA simulation. For the initial stage let $\sigma_0(\pi) = \text{SNEXT}(\pi, \lambda)$ for each position $\pi \in \Pi$. For $k = 0, 1, \dots, s_1(n) - 1$ and each $\pi, \pi' \in \Pi$ such that $h(\pi') \leq h(P_k(\pi))$ let $\sigma_k(\pi, \pi')$ be the stack derived from $\sigma_k(\pi)$ by $h(P_k(\pi)) - h(\pi')$ pops. Also, let $\sigma_{k+1}(\pi) = \sigma_k(\pi, Q_k(\pi)) \cdot \sigma_k(Q_k(\pi))$. (See Figure 8.)

We prove Theorem 5.1 by induction, using the following induction hypothesis for a fixed stage $k \geq 0$:

For each position $\pi \in \Pi$,

(a) $(P_k(\pi), \sigma_k(\pi)) \in \text{COMP}_{2^k + t(\pi)}(\pi, \lambda)$ (again, see Figure 8)

(b) $I_k(\pi)$ is that position $\pi' \in \text{PCOMP}(\pi, \lambda)$ such that $h(\pi') = h(\pi)$
 $t(\pi') \leq t(P_k(\pi))$ and $t(\pi')$ is maximal. (see Figure 9)

Also for each $\pi, \pi' \in \Pi$ such that $t(\pi') \geq t(P_k(\pi))$ and $h(P_k(\pi)) \geq h(\pi') \geq h(\pi)$ we assume in the induction hypothesis for stage k that $\text{PREDICT}_k(\pi, \pi') \in \text{SKIP}_{k, 2^k}(\pi, \pi')$ where for $\ell \geq 0$, $\text{SKIP}_{k, \ell}(\pi, \pi')$ is a position $\pi_1 \in \text{PCOMP}(\pi', \sigma_k(\pi, \pi'))$ such that $h(\pi_1) \leq h(\pi')$, $t(\pi_1) \geq t(P_k(\pi))$ and either $t(\pi_1) \geq t(\pi') + \ell$ or otherwise $t(\pi_1)$ is maximal and $\leq t(\pi') + \ell$. (I.e., there does not exist a position $\pi'' \in \text{PCOMP}(\pi', \sigma_k(\pi, \pi'))$ such that $h(\pi'') \leq h(\pi_1)$ and $t(\pi_1) < t(\pi'') < t(\pi') + \ell$.) See Figure 10.

The following lemma will be useful in extending the induction to stage $k+1$.

LEMMA 5.1. For any $\pi, \pi' \in \Pi$ such that $t(\pi') \geq t(P_k(\pi))$ and $h(P_k(\pi)) \geq h(\pi') \geq h(\pi)$, let $\pi_1 = \text{PREDICT}_k(\pi, \pi')$ and $\pi_2 = P_k(\pi_1)$. Then $\pi_2 \in \text{PCOMP}_{2^{k+t}(\pi')}$ and furthermore either $t(\pi_2) \geq t(\pi') + 2^k$ or $\pi_1 = \pi_2$.

Proof. By the induction hypothesis $\pi_1 \in \text{PCOMP}(\pi', \sigma_k(\pi, \pi'))$ where $t(\pi_1) \geq t(P_k(\pi))$ and $h(\pi_1) \leq h(\pi')$. Also by the induction hypothesis, $\pi_2 \in \text{PCOMP}_{2^{k+t}(\pi_1)}(\pi_1, \lambda)$. By Propositions 2.3 and 2.4, $\pi_2 \in \text{PCOMP}_{2^{k+t}(\pi')}(\pi', \sigma_k(\pi, \pi'))$. Suppose $\pi_1 \neq \pi_2$ and $t(\pi_2) < t(\pi') + 2^k$. Then by definition of PCOMP we must have $\text{LOOP}(\pi_2, \lambda)$, so $h(\pi_2) = h(\pi_1)$, by assumption A_3 of Section 2. But this violates the induction hypothesis of PREDICT_k . \square

LEMMA 5.2. For all $\pi \in \Pi$, $(P_{k+1}(\pi), \sigma_{k+1}(\pi)) \in \text{COMP}_{2^{k+1+t}(\pi)}(\pi, \lambda)$.

Proof. By the induction hypothesis, $(P_k(\pi), \sigma_k(\pi)) \in \text{COMP}_{2^{k+t}(\pi)}(\pi, \lambda)$. Note that $\sigma_k(\pi, P_k(\pi)) = \sigma_k(\pi)$. By Lemma 5.1, $P_{k+1}(\pi) \in \text{PCOMP}_{2^{k+t}(P_k(\pi))}(P_k(\pi), \sigma_k(\pi))$ and either $t(P_{k+1}(\pi)) \geq t(P_k(\pi)) + 2^k$ or $P_{k+1}(\pi) = Q_k(\pi)$. By Proposition 2.3, $P_{k+1}(\pi) \in \text{PCOMP}_{t_0}(\pi, \lambda)$, for $t_0 = 2^k + t(P_k(\pi))$. If $t(P_k(\pi)) \geq 2^k + t(\pi)$ then $t_0 \geq 2^{k+1} + t(\pi)$. Otherwise $\text{LOOP}(P_k(\pi), \lambda)$ so $P_{k+1}(\pi) = P_k(\pi)$ and so $\text{LOOP}(P_{k+1}(\pi), \lambda)$. Thus in either case $P_{k+1}(\pi) \in \text{PCOMP}_{2^{k+1+t}(\pi)}(\pi, \lambda)$. It is then easy to verify that the definition $\sigma_{k+1}(\pi) = \sigma_k(\pi, Q_k(\pi)) \circ \sigma_k(Q_k(\pi))$ satisfies $(P_{k+1}(\pi), \sigma_{k+1}(\pi)) \in \text{COMP}_{2^{k+1+t}(\pi)}(\pi, \lambda)$. \square

It is also easy to prove

LEMMA 5.3. For each $\pi \in \Pi$, $L_{k+1}(\pi)$ is that position $\pi' \in \text{PCOMP}(\pi, \lambda)$ such that $h(\pi') = 0$, $t(\pi') \leq t(P_{k+1}(\pi))$ and $t(\pi')$ is maximal.

Proof. Suppose not. Then the induction hypothesis for $L_k(\pi)$ or $L_k(Q_k(\pi))$ will be violated. \square

LEMMA 5.4. For $\pi, \pi' \in \Pi$ such that $t(\pi') \geq t(P_k(\pi))$ and $h(P_k(\pi)) \geq h(\pi') \geq h(\pi)$. $HOP_{k+1}(\pi, \pi') \in SKIP_{k, 2^{k+1}}(\pi, \pi')$.

Proof. Let $\pi_1 = PREDICT_k(\pi, \pi')$ and $\bar{\pi}_1 = Q_k(\pi_1)$. By the induction hypothesis, $\pi_1 \in SKIP_{k, 2^k}(\pi, \pi')$ so $h(\pi_1) \leq h(\pi')$, $t(\pi_1) > t(P_k(\pi))$, and $\neg \exists \pi'' \in PCOMP(\pi', \sigma_k(\pi, \pi'))$ such that $h(\pi'') \leq h(\pi_1)$ and $t(\pi_1) < t(\pi'') < t(\pi') + 2^k$. By Lemma 5.1, $P_k(\pi_1) \in PCOMP_{2^k + t(\pi')}(\pi', \sigma_k(\pi, \pi'))$ and either $t(P_k(\pi_1)) \geq t(\pi') + 2^k$ or $\pi_1 = P_k(\pi_1)$. By the induction hypothesis, $\bar{\pi}_1 \in SKIP_{k, 2^k}(\pi_1, P_k(\pi_1)) \subseteq PCOMP(P_k(\pi_1), \sigma_k(\pi_1))$, since $\sigma_k(\pi_1, P_k(\pi_1)) = \sigma_k(\pi_1)$. Also, by this induction hypothesis $h(\bar{\pi}_1) \leq h(P_k(\pi_1))$, $t(\bar{\pi}_1) \geq t(P_k(\pi_1))$, and $(\neg \exists \pi'' \in PCOMP(P_k(\pi_1), \sigma_k(\pi_1)))$ such that $h(\pi'') \leq h(\bar{\pi}_1)$ and $t(\bar{\pi}_1) < t(\pi'') < t(P_k(\pi)) + 2^k$. By Propositions 2.3 and 2.4, $\bar{\pi}_1 \in PCOMP(\pi', \sigma_k(\pi, \pi'))$.

First we consider the case $h(\bar{\pi}_1) > h(\pi_1)$; in this case $HOP_{k+1}(\pi, \pi') = L_k(\pi_1)$. But we already have $h(P_k(\pi_1)) \geq h(\bar{\pi}_1) > h(\pi_1)$ so by assumption A_3 , $LOOP(\pi_1, \lambda) = \text{false}$. Therefore $\pi_1 \neq P_k(\pi_1)$ so by Lemma 5.1, $t(P_k(\pi_1)) \geq t(\pi') + 2^k$. Thus by the induction hypothesis for $PREDICT_k$, $\neg \exists \pi'' \in PCOMP(\pi', \sigma_k(\pi, \pi'))$ such that $h(\pi'') \leq h(\bar{\pi}_1)$ and $t(\bar{\pi}_1) < t(\pi'') < t(\pi') + 2^{k+1}$.

This implies $\bar{\pi}_1 \in SKIP_{k, 2^{k+1}}(\pi, \pi')$.

Finally, we extend the induction to stage $k+1$ for $PREDICT_{k+1}$.

LEMMA 5.5. For all $\pi, \pi' \in \Pi$ such that $t(\pi') \geq t(P_{k+1}(\pi))$ and $h(P_{k+1}(\pi)) \geq h(\pi') \geq h(\pi)$, $PREDICT_{k+1}(\pi, \pi') \in SKIP_{k+1, 2^{k+1}}(\pi, \pi')$.

Proof. Let $\hat{\pi} = HOP_{k+1}(Q_k(\pi), \pi')$.

We first consider the case $h(\hat{\pi}) > h(Q_k(\pi))$, so $PREDICT_{k+1}(\pi, \pi') = \hat{\pi}$.
 If $\hat{\pi} \notin SKIP_{k+1, 2^{k+1}}(\pi, \pi')$ then it easily follows $\hat{\pi} \notin SKIP_{k, 2^{k+1}}(\pi, \pi')$
 contradicting Lemma 5.4.

On the other hand, consider the case $h(\hat{\pi}) = h(Q_k(\pi))$, so
 $PREDICT_{k+1}(\pi, \pi') = \hat{\pi}$ where $\hat{\pi} = HOP_{k+1}(\pi, \hat{\pi})$. If $\hat{\pi} \notin SKIP_{k+1, 2^{k+1}}(\pi, \pi')$
 then it follows that $\hat{\pi} \notin SKIP_{k, 2^{k+1}}(\pi, \hat{\pi})$, again contradicting Lemma 5.4.

6. SIMULATION OF P-RAMS BY DETERMINISTIC APDAs

THEOREM 6.1. *Let L be accepted by a deterministic $T(n)$ space bounded P-RAM. Then L is accepted by a space $T(n)$, $2^{O(T(n)^2)}$ time deterministic APDA.*

Proof. [Fortune and Wyllie, 78] prove in their Lemma 1b that L is accepted by a $T(n)^2$ space, $2^{O(T(n)^2)}$ time deterministic TM. We use exactly their algorithm, but implement it on an auxiliary deterministic pushdown machine. Their algorithm is recursive and requires a pushdown stack of size at most $T(n)$, where each element on the stack can be represented by a bit string of length $O(T(n))$. Thus only $T(n)$ space is required by our simulating deterministic APDA. \square

7. FURTHER RESULTS

7.1 Decreasing Processor Bounds

For deterministic CFLs, the $O(\log n)$ time recognition algorithm of Section 4 requires n^4 processors. This processor bound is decreased considerably in the full paper by applying a weighting to the priority of processor tasks which compute the PREDICT function, and recycling those processors below a useful priority (these are executing completed tasks). This technique is similar to a pebble weighting developed by [Braunmühl and Verbeek, 80] for sequential deterministic CFL recognition.

7.2 Implementation in a HMM

The algorithm of Section 4 can be implemented within the same time on a Hardware Modification Machine (HMM) of [Cook, 80], by introducing extra processors to update the information required to compute the PREDICT function. (Previously, [Dymond and Cook, 80] showed that HMMs accept in time $O(s(n))$ any language accepted by a deterministic TM with space $s(n) \geq \log n$.)

7.3 Extension to LR(k) Recognition

The LR(k) grammars defined in [Knuth, 65] are frequently used in practice for programming languages. Deterministic CFLs are exactly the languages with LR(1) grammars. It is easy to extend our parallel recognition algorithm for deterministic CFLs to recognize the languages of any LR(k) grammar, in time $O(\log n)$ by a slight modification of the initialization stage. (I.e., the next move function is modified to depend on k input symbols following the input head.)

REFERENCES

- Braunmühl, B. von and R. Verbeck, "A recognition algorithm for deterministic CFL's optimal in time and space," 21st IEEE Symposium on Foundations of Computer Science, Syracuse, New York, pp. 411-420, (1980).
- Cook, S. A., "Deterministic CFL's are accepted simultaneously in polynomial time and log squared space," Proceedings of the 11th ACM Symposium on Theory of Computing, pp 338-345, (1979).
- Cook, S. A., "Towards a complexity theory of synchronous parallel computation," Presented at *Internationales Symposium über Logik und Algorithmik zu Ehren von Professor Hort Specker*, Zürich, Switzerland, February 1980.
- Dymond, P. and S. A. Cook, "Hardware complexity and parallel computation," IEEE FOCS Conference, 1980.
- Early, J., "An efficient context-free parsing algorithm," *Comm. ACM* 13:2, pp. 94-102, (1970).
- Fortune, S. and J. Wyllie, "Parallelism in random access machines," Proc. of the 10th ACM Symposium on Theory of Computation, pp. 114-118, (1970).
- Hopcroft, J. E. and J. D. Ullman, *Introduction to Automata, Theory, Languages and Computation*, Addison-Wesley, 1979.
- Knuth, D. E., "On the translation of languages from left to right," *Information and Control*, 8, 6, pp. 607-639, (1965).
- Lewis, P. M., R. E. Stearns and J. Hartmanis, "Memory bounds for recognition of context-free and context sensitive languages," IEEE Conference Record on Switching Theory and Logical Design, pp. 191-202, (1965).
- Ruzzo, W. L., "On uniform circuit complexity," Proceedings of the 20th IEEE Symposium on Foundations of Computer Science, pp. 312-318, (1979).
- Wyllie, J., "The complexity of parallel computation," Ph.D. Thesis, Cornell University, 1979.

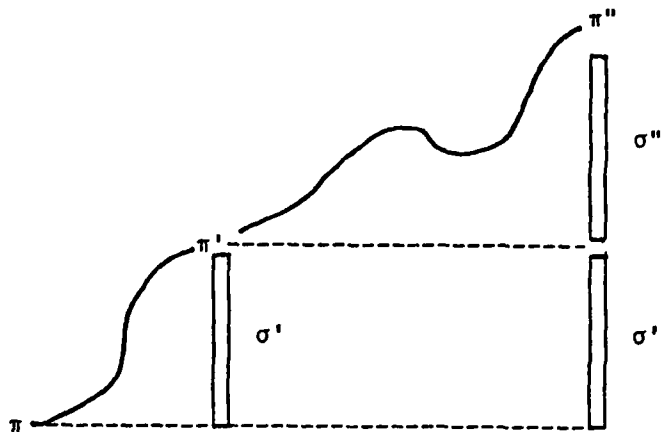


Figure 1. Combining APDA computations by Proposition 2.3.

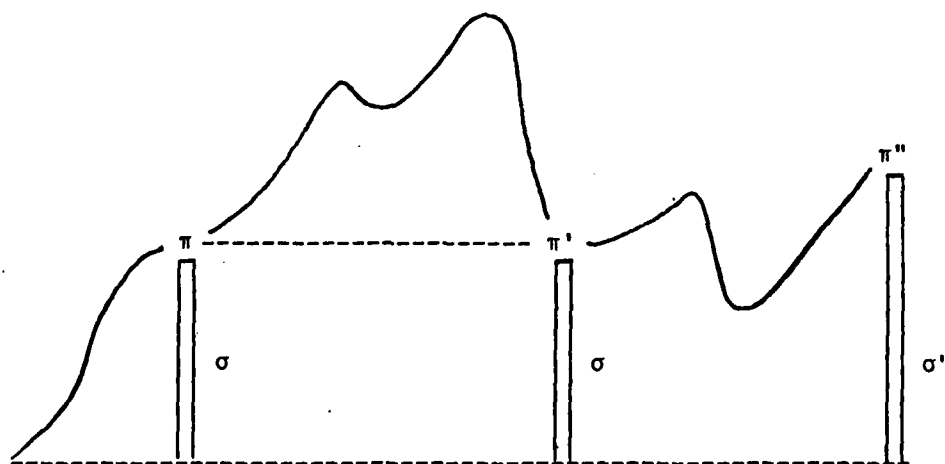


Figure 2. Combining APDA computation by Proposition 2.4.

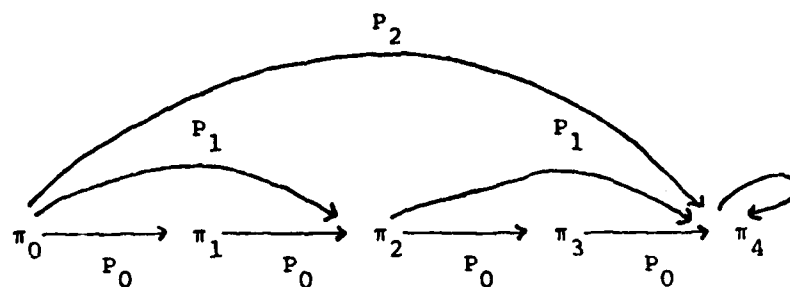


Figure 3. Examples of P_0, P_1 and P_2 .

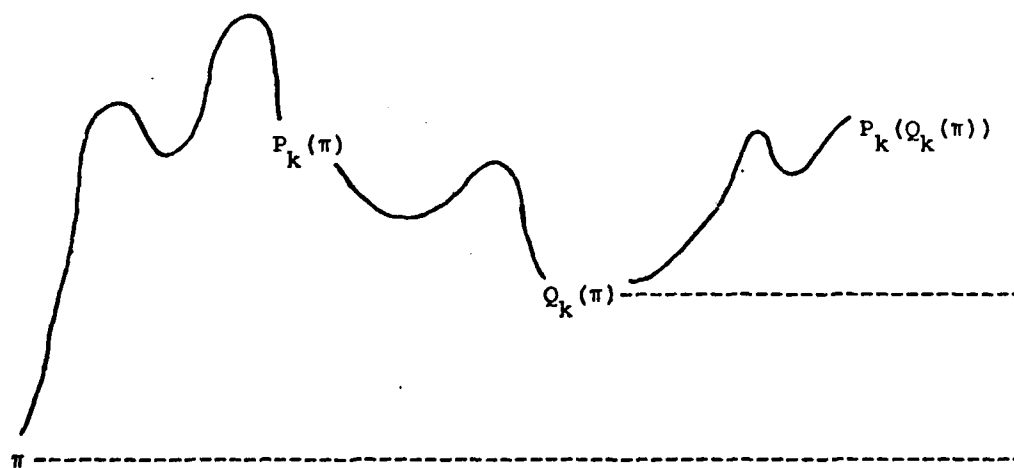


Figure 4. Example of $P_{k+1}(\pi) = P_k(Q_k(\pi))$

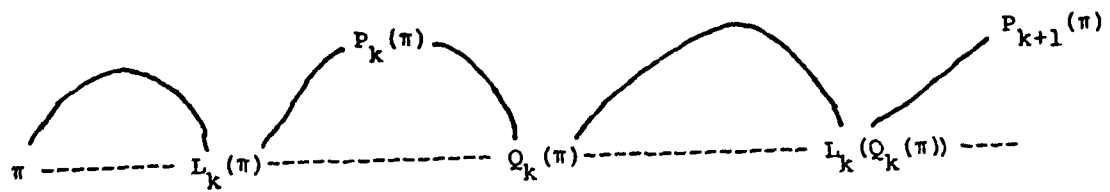


Figure 5a. $L_{k+1}(\pi) = L_k(Q_k(\pi))$ for case $h(Q_k(\pi)) = h(\pi)$.

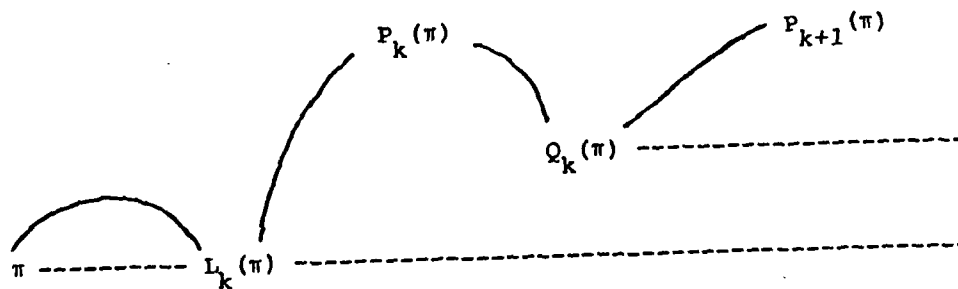


Figure 5b. $L_{k+1}(\pi) = L_k(\pi)$ for case $h(Q_k(\pi)) > h(\pi)$.

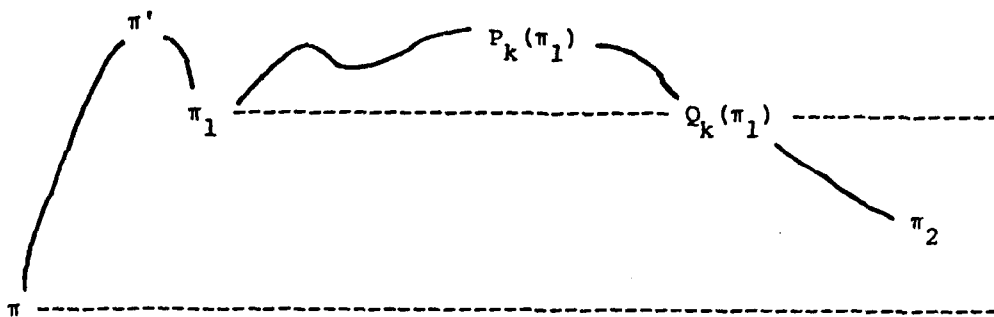


Figure 6a. $\text{HOP}_{k+1}(\pi, \pi') = \pi_2$ in the case $h(Q_k(\pi_1)) = h(\pi_1)$.

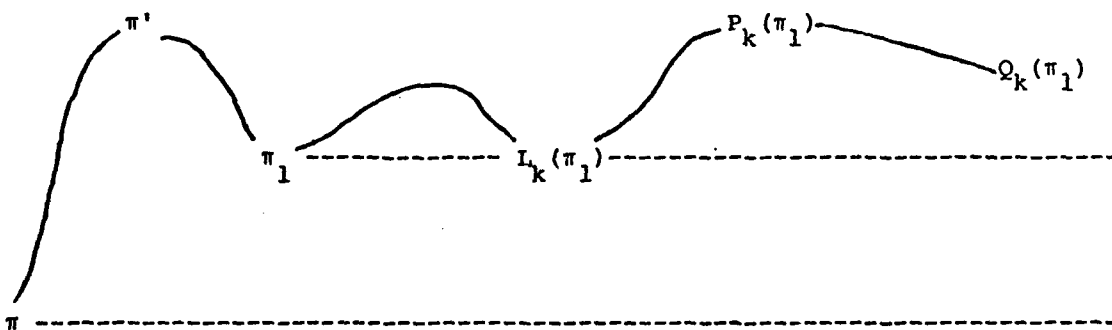


Figure 6b. $\text{HOP}_{k+1}(\pi, \pi') = L_k(\pi_1)$ in the case $h(Q_k(\pi_1)) > h(\pi_1)$.

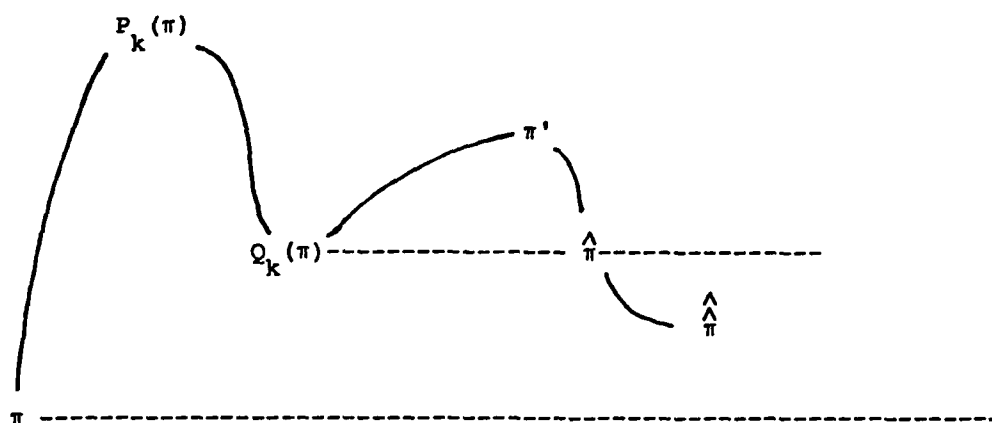


Figure 7a. $\text{PREDICT}_{k+1}(\pi, \pi') = \hat{\pi}$ in the case $h(\hat{\pi}) = h(Q_k(\pi))$.

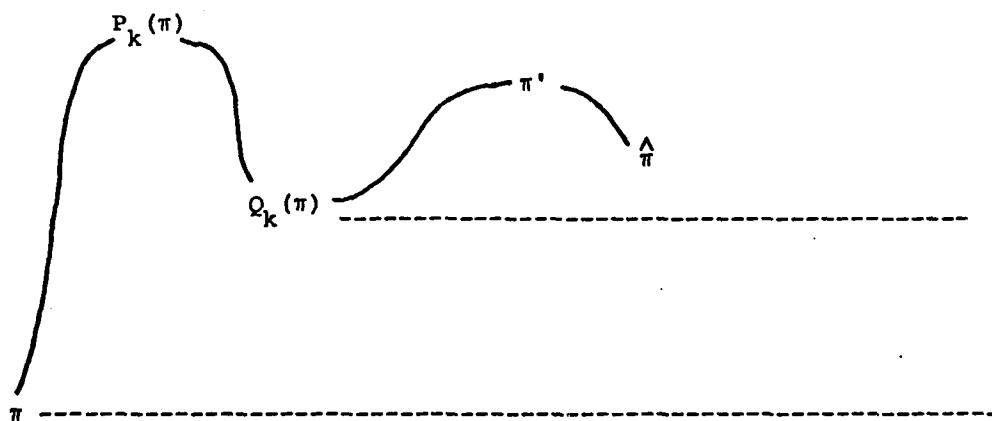


Figure 7b. $\text{PREDICT}_{k+1}(\pi, \pi') = \hat{\pi}$ in the case $h(\hat{\pi}) > h(Q_k(\pi))$.

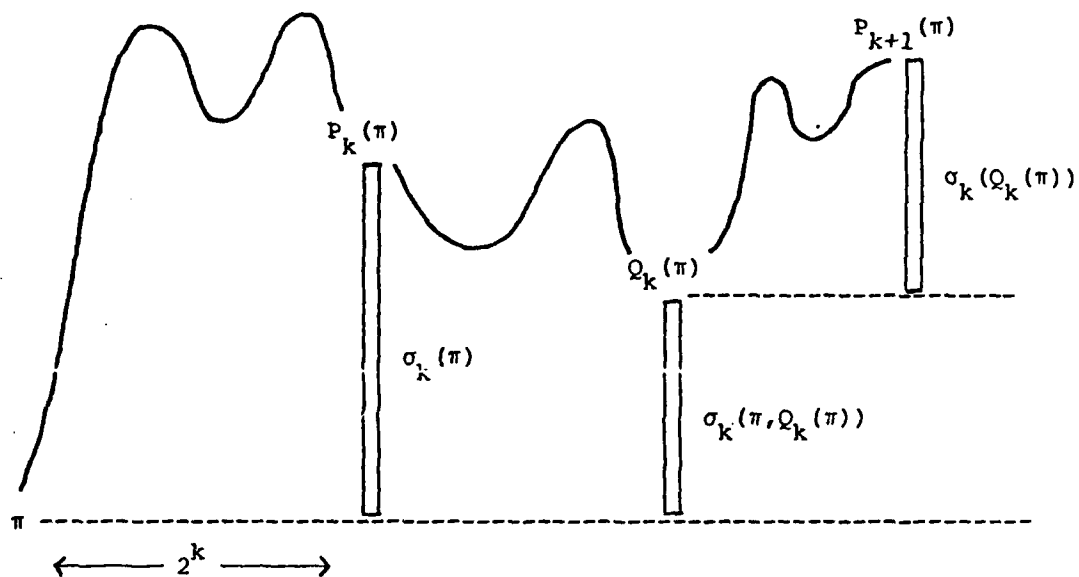


Figure 8. Example of definition of $\sigma_{k+1}(\pi)$ for computation given in Figure 4.

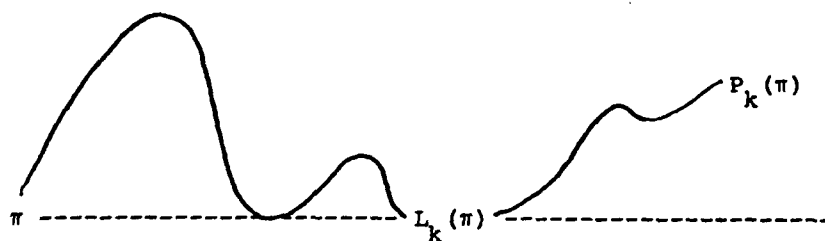


Figure 9. Induction hypothesis for $L_k(\pi)$.

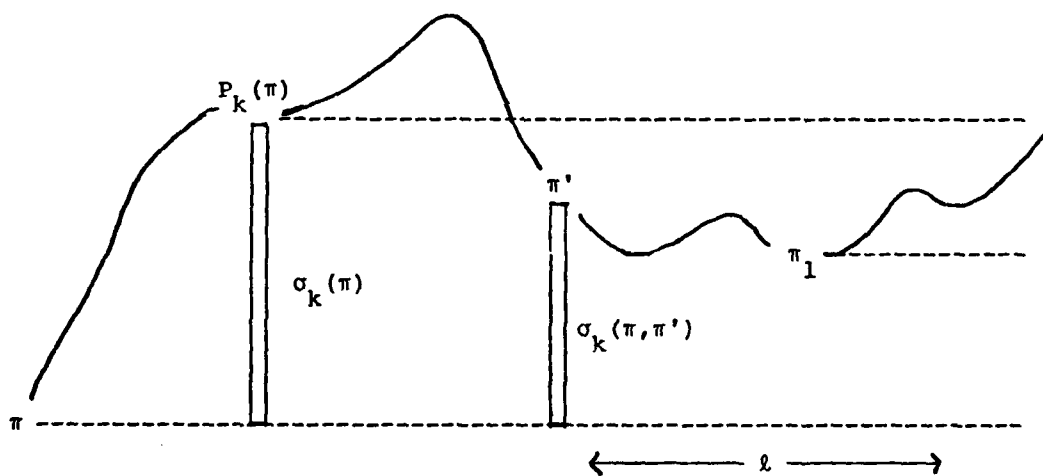


Figure 10. An example of definition of $SKIP_{k,\ell}(\pi, \pi') = \pi_1$.

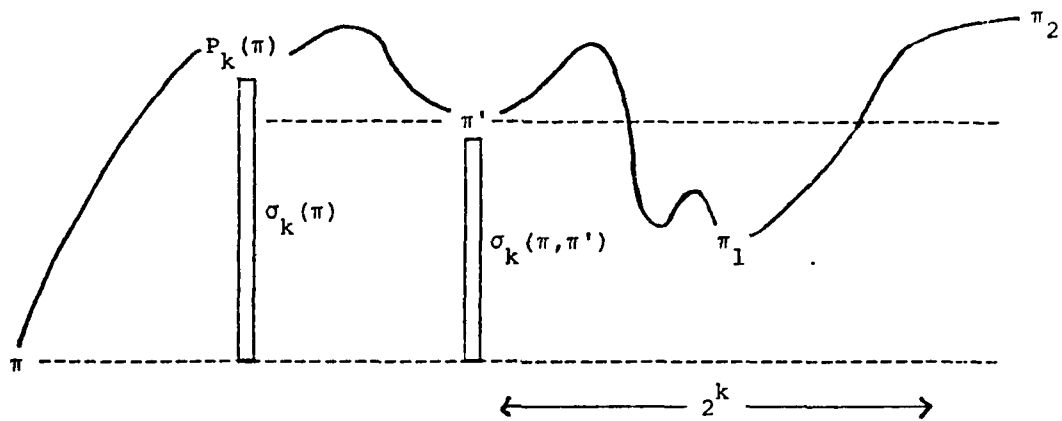


Figure 11. An example of Lemma 5.1, for the case $t(\pi_2) \geq t(\pi') + 2^k$.

**ATE
MED**